```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

typedef struct Node Node;

int compare(const void* a, const void* b) {
    return (*(int*)a - *(int*)b);
}

void append(Node** head, int data) {
    Node* new_node = (Node*)malloc(sizeof(Node));
    new_node->data = data;
    new_node->next = NULL;

    if (*head == NULL) {
        *head = new_node;
        return;
    }

    Node* last_node = *head;
    while (last_node->next != NULL) {
        last_node = last_node->next;
    }
```

```
    last_node->next = new_node;
}

void display(Node* head) {
    Node* current = head;
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}

void sort_list(Node** head) {
    if (*head == NULL || (*head)->next == NULL) {
        return;
    }

    Node* current = *head;
    int values[100], count = 0;

    while (current != NULL) {
        values[count++] = current->data;
        current = current->next;
    }

    qsort(values, count, sizeof(int), compare);

    current = *head;
    for (int i = 0; i < count; i++) {
```

```
        current->data = values[i];
        current = current->next;
    }
}

void reverse_list(Node** head) {
    Node* prev = NULL;
    Node* current = *head;
    Node* next_node;

    while (current != NULL) {
        next_node = current->next;
        current->next = prev;
        prev = current;
        current = next_node;
    }

    *head = prev;
}

void concatenate(Node** head1, Node* head2) {
    if (*head1 == NULL) {
        *head1 = head2;
    } else {
        Node* current = *head1;
        while (current->next != NULL) {
            current = current->next;
        }
        current->next = head2;
```

```
    }
}

int main() {
    Node* list1 = NULL;
    Node* list2 = NULL;

    // Append elements to list1
    append(&list1, 3);
    append(&list1, 1);
    append(&list1, 4);

    // Append elements to list2
    append(&list2, 2);
    append(&list2, 5);
    append(&list2, 6);

    // Display original lists
    printf("Original List 1:\n");
    display(list1);

    printf("\nOriginal List 2:\n");
    display(list2);

    // Sort and display
    sort_list(&list1);
    printf("\nSorted List 1:\n");
    display(list1);
```

```
    // Reverse and display
    reverse_list(&list1);
    printf("\nReversed List 1:\n");
    display(list1);

    // Concatenate and display
    concatenate(&list1, list2);
    printf("\nConcatenated Lists:\n");
    display(list1);

    return 0;
}
```